EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	4358	((replication or copy) with (sync or synchronization))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:49
L2	484	(dynamic with (copy or replication)) and ((thread$1 or process$3) with (concurrent or parallel))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:56
L3	83	L1 and L2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:49
L4	63	L3 and @ad<"20030527"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:31
L5	42	L4 and kernel and @ad<"20030527"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:51
L6	9	L4 and kernel and @ad<"20030527" and (707/200-204).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:51
L7	56	(dynamic with (copy or replicat$3)) and ((thread$1 or process$2) with (concurrent or parallel)) and (synchronization with (copy or replicat$3))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:30
L8	50	7 and @ad<"20030527"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:31

EAST Search History

L9	37	8 and queue	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:33
L10	1	9 and (wait$3 with state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:32
S1	3	(synchronization with thread) and (dynamic with (copy or replication)) and (thread with (concurrent or parallel))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:38
S2	5	(synchronization with thread) and (dynamic with (copy or replication)) and (thread$1 with (concurrent or parallel))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:57
S3	137	(synchronization with (thread or process$3)) and (dynamic with (copy or replication)) and ((thread$1 or process$3) with (concurrent or parallel))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:48
S4	13	S3 and (process$3 with kernel)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:50
S5	10	S4 and @ad<"20030527"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:57
S6	50	S3 and (synchronization with command)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:51

EAST Search History

S7	1	S6 and (replication with command)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:51
S8	0	S6 and ((replication or copy) with sychronization)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:52
S9	8	S6 and ((replication or copy) with (sync or synchronization))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:56
S10	4358	((replication or copy) with (sync or synchronization))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:56
S11	484	(dynamic with (copy or replication)) and ((thread$1 or process$3) with (concurrent or parallel))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:57
S12	83	S10 and S11	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 10:57
S13	63	S12 and @ad<"20030527"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 12:49
S14	15	S13 and (synchronization with (concurrent or parallel) with (thread$1 or process$3))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 11:10

EAST Search History

S15	40	S13 and queue	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:32
S16	3	S15 and (wait$3 with state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 13:32
S17	10	S13 and ((sync or synchronization) with command)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/10 11:13

Web Results **1 - 10** of about **372,000** for **synchronization thread dynamic replication concurrent**. (**0.30** secon

[PDF] **Dynamic Replication** and **Synchronization** of Web Services for High ...
File Format: PDF/Adobe Acrobat - View as HTML
Dynamic Replication and **Synchronization** of Web Services. 7. } wait until **threads** finish. // exchange data with other monitors. start **concurrent threads** for ...
www.infosys.tuwien.ac.at/staff/lukasz/dustdar_juszczyk_SOCA.pdf - Similar pages

[PDF] **Dynamic replication** and **synchronization** of web services for high ...
File Format: PDF/Adobe Acrobat
Dynamic replication and **synchronization** of web services. for high availability in mobile ad-hoc ..... start **concurrent threads** for all hosts in monitors { ...
www.springerlink.com/index/H16750028121U656.pdf - Similar pages

[PDF] Eliminating **Synchronization** Bottlenecks Using Adaptive **Replication**
File Format: PDF/Adobe Acrobat - View as HTML
Synchronization bottlenecks occur when multiple **threads** attempt to ..... simplifies the application of **dynamic replication** to object-based programs, it is ...
www.cag.lcs.mit.edu/~rinard/paper/toplas03Adaptive**Replication**.pdf - Similar pages

[PS] Eliminating **Synchronization** Bottlenecks in Object-Based Programs ...
File Format: Adobe PostScript - View as Text
threads. **Synchronization** bottlenecks show up as a large amount .... ysis and **replication** for **concurrent** read access in shared memory. systems. ...
www.cag.lcs.mit.edu/~rinard/paper/ics99.ps - Similar pages
[More results from www.cag.lcs.mit.edu]

Team-Regal:Safe **concurrent** programming
New Results - Safe **concurrent** programming. ... reconciliation and commitment · Safe **concurrent** programming · **Dynamic replication** in multi-agent systems ...
ralyx.inria.fr/2006/Raweb/regal/uid56.html - 10k - Cached - Similar pages

Kan Motivation
Programming errors caused by incorrect **thread synchronization** and invalid ... of a global name space in which objects are accessed by **concurrent threads**. ...
www.ittc.ku.edu/kan/motivation.html - 13k - Cached - Similar pages

Kan Documents
Thread Optimizations in **Concurrent** Object Oriented Languages.... We use caching and **dynamic replication** to resolve this bottleneck. ...
www.ittc.ku.edu/kan/documents/ - 15k - Cached - Similar pages

[PDF] **Concurrent Replication** of Parallel and Distributed Simulations
File Format: PDF/Adobe Acrobat
simulation framework supporting **Concurrent Replication** ..... and **synchronization** in distributed simulations have been. widely analyzed and discussed in the ...
ieeexplore.ieee.org/iel5/9855/31040/01443329.pdf?arnumber=1443329 - Similar pages

Eliminating **synchronization** bottlenecks using adaptive **replication**
Synchronization bottlenecks occur when multiple **threads** attempt to concurrently ... 13 Damien Doligez , Xavier Leroy, A **concurrent**, generational garbage ...
portal.acm.org/citation.cfm?id=641911&dl=ACM&coll=portal - Similar pages

Concurrent Replication of Parallel and Distributed Simulations ...
Slide 24: Conclusions: **Concurrent Replication** of PADS Under the Parallel (or ... the new multimedia format; **sync** your slides to audio (any mp3 URL) ...
www.slideshare.net/gda/**concurrent-replication**-of-parallel-and-distributed-simulations - 77k - Cached - Similar pages

1 2 3 4 5 6 7 8 9 10 **Next**

---

synchronization thread dynamic repl Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---

Terms used:
synchronization thread dynamic replication concurrent

Found **31,209** of **207,474**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Eliminating synchronization bottlenecks using adaptive replication

Martin C. Rinard, Pedro C. Diniz

May 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 3

Publisher: ACM Press

Full text available: pdf(826.28 KB) Additional Information: full citation, abstract, references, index terms, review

This article presents a new technique, adaptive replication, for automatically eliminating synchronization bottlenecks in multithreaded programs that perform atomic operations on objects. Synchronization bottlenecks occur when multiple threads attempt to concurrently update the same object. It is often possible to eliminate synchronization bottlenecks by replicating objects. Each thread can then update its own local replica without synchronization and without interacting with other threads. When ...

Keywords: Atomic operations, commutativity analysis, parallel computing, parallelizing compilers, replication, synchronization

2 Concurrent Replication of Parallel and Distributed Simulations

Luciano Bononi, Michele Bracuto, Gabriele D'Angelo, Lorenzo Donatiello

June 2005 **Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation PADS '05**

Publisher: IEEE Computer Society

Full text available: pdf(157.01 KB) Additional Information: full citation, abstract, index terms

Parallel and distributed simulations enable the analysis of complex systems by concurrently exploiting the aggregate computation power and memory of clusters of execution units. In this paper we investigate a new direction for increasing both the speedup of a simulation process and the utilization of computation and communication resources. Many simulation-based investigations require to collect independent observations for a correct and significant statistical analysis of results. The execution ...

3 Eliminating synchronization bottlenecks in object-based programs using adaptive replication

Martin Rinard, Pedro Diniz

May 1999 **Proceedings of the 13th international conference on Supercomputing ICS '99**

Publisher: ACM Press
Full text available: pdf(1.27 MB) Additional Information: full citation, references, citings, index terms

4 Concurrent replicating garbage collection
James O'Toole, Scott Nettles
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming LFP '94**, Volume VII Issue 3
Publisher: ACM Press
Full text available: pdf(919.87 KB) Additional Information: full citation, abstract, references, citings, index terms

We have implemented a concurrent copying garbage collector that uses replicating garbage collection. In our design, the client can continuously access the heap during garbage collection. No low-level synchronization between the client and the garbage collector is required on individual object operations. The garbage collector replicates live heap objects and periodically synchronizes with the client to obtain the client's current root set and mutation log. An experimental implementation usi ...

5 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research CASCON '97**
Publisher: IBM Press
Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

6 Concurrency and distribution in object-oriented programming
Jean-Pierre Briot, Rachid Guerraoui, Klaus-Peter Lohr
September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3
Publisher: ACM Press
Full text available: pdf(289.34 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper aims at discussing and classifying the various ways in which the object paradigm is used in concurrent and distributed contexts. We distinguish among the library approach, the integrative approach, and the reflective approach. The library approach applies object-oriented concepts, as they are, to structure concurrent and distributed systems through class libraries. The integrative approach consists of merging concepts such as obj ...

Keywords: concurrency, distribution, integration, libraries, message passing, object, reflection

7 The STAMPede approach to thread-level speculation
J. Gregory Steffan, Christopher Colohan, Antonia Zhai, Todd C. Mowry
August 2005 **ACM Transactions on Computer Systems (TOCS)**, Volume 23 Issue 3
Publisher: ACM Press
Additional Information: full citation, abstract, references, citings, index

Full text available: pdf(1.72 MB) terms

Multithreaded processor architectures are becoming increasingly commonplace: many current and upcoming designs support chip multiprocessing, simultaneous multithreading, or both. While it is relatively straightforward to use these architectures to improve the throughput of a multithreaded or multiprogrammed workload, the real challenge is how to easily create *parallel software* to allow single programs to effectively exploit all of this raw performance potential. One promising technique fo ...

Keywords: Thread-level speculation, automatic parallelization, cache coherence, chip-multiprocessing

8 Very concurrent mark-&-sweep garbage collection without fine-grain synchronization

Lorenz Huelsbergen, Phil Winterbottom
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3
Publisher: ACM Press

Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a new incremental algorithm for the concurrent reclamation of a program's allocated, yet unreachable, data. Our algorithm is a variant of mark-&-sweep collection that---unlike prior designs---runs mutator, marker, and sweeper threads concurrently *without* explicit fine-grain synchronization on shared-memory multiprocessors. A global, but infrequent, synchronization coordinates the per-object coloring marks used by the three threads; fine-grain synchronization is achieve ...

9 Models and languages for parallel computation

David B. Skillicorn, Domenico Talia
June 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 2
Publisher: ACM Press

Full text available: pdf(298.05 KB) Additional Information: full citation, abstract, references, citings, index terms

We survey parallel programming models and languages using six criteria to assess their suitability for realistic portable parallel programming. We argue that an ideal model should by easy to program, should have a software development methodology, should be architecture-independent, should be easy to understand, should guarantee performance, and should provide accurate information about the cost of programs. These criteria reflect our belief that developments in parallelism must be driven b ...

Keywords: general-purpose parallel computation, logic programming languages, object-oriented languages, parallel programming languages, parallel programming models, software development methods, taxonomy

10 A parallel, incremental, mostly concurrent garbage collector for servers

Katherine Barabash, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Yoav Ossia, Avi Owshanko, Erez Petrank
November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6
Publisher: ACM Press

Full text available: pdf(683.50 KB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded applications with multigigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC

include: ensuring short pause times on a multigigabyte heap while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multicycle fence instructions required by weak ordering to a minimum.We designed and implemented a collector facing these demands building on th ...

Keywords: Garbage collection, JVM, concurrent garbage collection

11 Sapphire: copying GC without stopping the world

Richard L. Hudson, J. Eliot B. Moss
June 2001 **Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande JGI '01**
Publisher: ACM Press

Full text available: pdf(899.45 KB) Additional Information: full citation, abstract, references, citings, index terms

Many concurrent garbage collection (GC) algorithms have been devised, but few have been implemented and evaluated, particularly for the Java programming language. Sapphire is an algorithm we have devised for concurrent copying GC. Sapphire stresses minimizing the amount of time any given application thread may need to block to support the collector. In particular, Sapphire is intended to work well in the presence of a large number of application threads, on small- to medium-scale shared memor ...

12 Concurrent garbage collection using hardware-assisted profiling

Timothy H. Heil, James E. Smith
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1
Publisher: ACM Press
Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, citings, index terms

In the presence of on-chip multithreading, a Virtual Machine (VM) implementation can readily take advantage of *service threads* for enhancing performance by performing tasks such as profile collection and analysis, dynamic optimization, and garbage collection concurrently with program execution. In this context, a hardware-assisted profiling mechanism is proposed. The *Relational Profiling Architecture* (RPA) is designed from the top down. RPA is based on a relational model similar ...

13 JR: Flexible distributed programming in an extended Java

Aaron W. Keen, Tingjian Ge, Justin T. Maris, Ronald A. Olsson
May 2004 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 26 Issue 3
Publisher: ACM Press
Full text available: pdf(167.00 KB) Additional Information: full citation, abstract, references, index terms

Java provides a clean object-oriented programming model and allows for inherently system-independent programs. Unfortunately, Java has a limited concurrency model, providing only threads and remote method invocation (RMI).The JR programming language extends Java to provide a rich concurrency model, based on that of SR. JR provides dynamic remote virtual machine creation, dynamic remote object creation, remote method invocation, asynchronous communication, rendezvous, and dynamic process creation ...

Keywords: Concurrency, Java, SR, concurrent object-oriented programming

14 The family of concurrent logic programming languages

Ehud Shapiro

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Publisher: ACM Press

Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

15 Distributed systems - programming and management: On remote procedure call
Patrícia Gomes Soares
November 1992 **Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 2 CASCON '92**
Publisher: IBM Press
Full text available: pdf(4.52 MB) Additional Information: full citation, abstract, references, citings

The Remote Procedure Call (RPC) paradigm is reviewed. The concept is described, along with the backbone structure of the mechanisms that support it. An overview of works in supporting these mechanisms is discussed. Extensions to the paradigm that have been proposed to enlarge its suitability, are studied. The main contributions of this paper are a standard view and classification of RPC mechanisms according to different perspectives, and a snapshot of the paradigm in use today and of goals for t ...

16 The development of the Emerald programming language
Andrew P. Black, Norman C. Hutchinson, Eric Jul, Henry M. Levy
June 2007 **Proceedings of the third ACM SIGPLAN conference on History of programming languages HOPL III**
Publisher: ACM Press
Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, index terms

Emerald is an object-based programming language and system designed and implemented in the Department of Computer Science at the University of Washington in the early and mid-1980s. The goal of Emerald was to simplify the construction of distributed applications. This goal was reflected at every level of the system: its object structure, the programming language design, the compiler implementation, and the run-time support.

This paper describes the origins of the Emerald group, the for ...

Keywords: Eden, Emerald, Washington, abstract types, call-by-move, distributed programming, mobility, object mobility, object-oriented programming, polymorphism, remote object invocation, remote procedure call, type conformity

17 Programming languages for distributed computing systems
Henri E. Bal, Jennifer G. Steiner, Andrew S. Tanenbaum
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Publisher: ACM Press

Full text available: pdf(6.50 MB) Additional Information: full citation, abstract, references, citings, index terms, review

When distributed systems first appeared, they were programmed in traditional sequential languages, usually with the addition of a few library procedures for sending and receiving

messages. As distributed applications became more commonplace and more sophisticated, this ad hoc approach became less satisfactory. Researchers all over the world began designing new programming languages specifically for implementing distributed applications. These languages and their history, their underlying pr ...

18 Demonstrating the scalability of a molecular dynamics application on a Petaflop computer

George S. Almasi, C□lin Caşcaval, José G. Castaños, Monty Denneau, Wilm Donath, Maria Eleftheriou, Mark Giampapa, Howard Ho, Derek Lieber, José E. Moreira, Dennis Newns, Marc Snir, Henry S. Warren

June 2001 **Proceedings of the 15th international conference on Supercomputing ICS '01**

Publisher: ACM Press

Full text available: pdf(392.72 KB) Additional Information: full citation, abstract, references, citings, index terms

The IBM Blue Gene project has endeavored into the development of a cellular architecture computer with millions of concurrent threads of execution. One of the major challenges of this project is demonstrating that applications can successfully exploit this massive amount of parallelism. Starting from the sequential version of a well known molecular dynamics code, we developed a new application that exploits the multiple levels of parallelism in the Blue Gene cellular architecture. We perform ...

Keywords: Blue Gene, cellular architecture, massively parallel computing, molecular dynamics, performance evaluation

19 A machine independent interface for lightweight threads

Bodhisattwa Mukherjee, Greg Eisenhauer, Kaushik Ghosh

January 1994 **ACM SIGOPS Operating Systems Review**, Volume 28 Issue 1

Publisher: ACM Press

Full text available: pdf(840.43 KB) Additional Information: full citation, abstract, citings, index terms

Recently, lightweight thread libraries have become a common entity to support concurrent programming on shared memory multiprocessors. However, the disparity between primitives offered by operating systems creates a challenge for those who wish to create portable lightweight thread packages. What should be the interface between the machine-independent and machine-dependent parts of the thread library? We have implemented a portable lightweight thread library on top of Unix on a KSR-1 supercomput ...

20 Software behavior oriented parallelization

Chen Ding, Xipeng Shen, Kirk Kelsey, Chris Tice, Ruke Huang, Chengliang Zhang

June 2007 **ACM SIGPLAN Notices , Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation PLDI '07**, Volume 42 Issue 6

Publisher: ACM Press

Full text available: pdf(299.09 KB) Additional Information: full citation, abstract, references, index terms

Many sequential applications are difficult to parallelize because of unpredictable control flow, indirect data access, and input-dependent parallelism. These difficulties led us to build a software system for behavior oriented parallelization (BOP), which allows a program to be parallelized based on partial information about program behavior, for example, a user reading just part of the source code, or a profiling tool examining merely one or few executions.

The basis of BOP is program ...

Keywords: program behavior, speculative parallelization

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Results for **"(synchronization thread dynamic replication concurrent) <in> metadata"**
Your search matched **0** of **1628575** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

e-mail

Login

Username

Password

» Forgot your password?

Please remember to log out when you have finished your session.

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistan search.

» Key

	Indicates full text access
IEEE JNL	IEEE Journal or Magazine
IET JNL	IET Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IET CNF	IET Conference Proceeding
IEEE STD	IEEE Standard